Project F6 - House Prices: Advanced Regression Techniques

Course: Introduction to Data Science (LTAT.02.002), Spring 2025
Team: Leonid Tšigrinski, Kirsika Roos-Nessler, Jessica Brjuhhov
Repository: https://github.com/leonid-98/uni-data-science-project-2025

Task 1

The GitHub repository `uni-data-science-project-2025` hosts all project assets, and invitations have been sent to the instructors. Project initialization covers environment pinning in `src/setup_env.ipynb` and verification of the Kaggle API token. This setup lets us run as many Kaggle-friendly experiments as we want without fighting configuration issues.

Task 2

Identifying the business goals

Background:
Our project tackles the Kaggle "House Prices: Advanced Regression Techniques" challenge. The dataset collects structural, material, and location-related features for homes in Ames, Iowa. Because our team is still building ML experience, we treat the task as "try everything we know and see what works best." Since it is a Kaggle competition, we also want to demonstrate one or two ideas that are not already posted.

Business goals:
We want to build a regression workflow that predicts house prices with a root mean squared error (RMSE). The short-term goal is to beat the Kaggle baseline notebook and understand why each model variant performs the way it does. The medium-term goal is to document a modeling recipe that future students can reuse for similar Kaggle competitions.

Business success criteria:
The best-case scenario is to top the public leaderboard. We will aim for that, but the primary goal is to deliver a fresh approach and gain deeper understanding of the ML workflow.

Assessing the situation

Inventory of resources:
We already have local clones of the Kaggle dataset, Python 3 environments, and shared notebooks in Git. Most experiments will run locally. If we hit compute limits, we can fall back to the UT HPC cluster (through another course) or the Google Colab free-tier GPUs.

Requirements, assumptions, and constraints:
The feature set is fixed by Kaggle, so we do not collect new data. Competition rules forbid sharing the CSV files, which is why the repository only stores notebooks, code, and reports. All experiments must run with reproducible seeds so we can compare model families fairly.

Risks and contingencies

Main risks include:

1. Spending too much time on weak ideas
2. Data leakage from careless feature engineering
3. Overfitting when stacking aggressive models

To reduce these risks we will log every experiment, keep a validation diary, and reserve the last few days for focused tuning of the two most promising approaches.

Terminology:
"SalePrice" is the Kaggle target (log-transformed for scoring).
"Leaderboards" refer to Kaggle's public/private splits.
"Blends" describe simple averages of multiple model predictions.

Costs and benefits:
Direct costs are zero because Kaggle and our tools are free. The benefit to our imaginary stakeholder is higher confidence when listing houses. The benefit to us is hands-on practice with CRISP-DM, model comparison, and teamwork under a deadline.

Defining your data-mining goals

Data-mining goals:

Build a modeling pipeline that:

1. Loads the Kaggle training data without errors
2. Handles imputations and encoding consistently between training and inference
3. Experiments with several model families (regularized linear models, tree ensembles, boosting variants, and blends)
4. Logs Kaggle leaderboard feedback for each submission so we learn which ideas help

Data-mining success criteria:
We target cross-validated log-RMSE below 0.135 and Kaggle public log-RMSE below 0.14. Generally, we want at least three distinct modeling approaches documented with notes on why they worked or failed. Recording SHAP or permutation importances for the final candidates will help us explain the results to classmates and instructors.

Task 3. Data understanding

Gathering data

Outline data requirements. We need all rows from `data/train.csv` for supervised learning and the unlabeled rows in `data/test.csv` for generating Kaggle submissions. Essential fields include quantitative measures (e.g., `GrLivArea`, `LotArea`), categorical descriptors (e.g., `Neighborhood`, `Exterior1st`), and date-related indicators such as `YrSold`. Supporting metadata from `data/data_description.txt` explains each field's semantics and will be referenced when engineering features.

Verify data availability. The Kaggle download bundle, accessed via the competition API, already resides in `data/`. We confirmed that `data/train.csv`, `data/test.csv`, and `data/sample_submission.csv` match the file sizes listed on the competition page, and `setup_env.ipynb` logs success after each download. Each team member configured `~/.kaggle/kaggle.json` and validated access with `kaggle competitions files -c house-prices-advanced-regression-techniques`.

Define selection criteria. We plan to use every labeled row in `data/train.csv`. Feature drops will only happen if a variable adds noise or duplicates information, and we will keep notes when that occurs. External data sources are off the table.

Describing data

Descriptive summaries drawn from `data/train.csv` show that numerical features vary widely (e.g., `LotArea` ranges from 1300 to 215000 square feet), while categorical levels such as `Neighborhood` present moderately imbalanced distributions dominated by "NAmes" and "CollgCr". Year-based attributes depict housing waves between 1950-2010, aligning with Ames development history. Target `SalePrice` has a right-skewed distribution; log-transforming it yields a near-normal shape, justifying the competition's metric choice.

Exploring data

Early correlation checks already show that `OverallQual` and `GrLivArea` rise with `SalePrice`, while basement features only matter after a certain size. We also spotted a handful of massive homes (over 4000 sq ft) that may skew results, so they will be flagged before training. Full plots are still on the to-do list, but these first passes tell us which transformations to try when we iterate through different models.

Verifying data quality

Missing values mainly occur in optional amenities such as `PoolQC`, `Alley`, and `Fence`, and the documentation says many of those "NA" entries simply mean "feature missing on site." We will keep them as separate categories and fill numeric gaps like `LotFrontage` with neighborhood medians. The IDs in `data/train.csv` are unique, cross-field checks like `GarageCars` vs `GarageArea` look consistent, and `data/sample_submission.csv` shows the `Id` / `SalePrice` order we must follow. Overall, the data is clean enough to support multiple modeling pipelines once we handle these few gaps and outliers.

Task 4

Task allocation and hours.

Task	Description	Leonid (h)	Kirsika (h)	Jessica (h)
T1	Environment setup, Kaggle API checks, repo hygiene	4	2	2
T2	Data cleaning, feature audits, and plotting (EDA deck)	3	8	4
T3	Feature engineering & preprocessing pipelines	6	6	3
T4	Modeling, ensemblings and hyperparameter search	8	4	4
T5	Validation runs, submission packaging, leaderboard tracking	6	4	5
T6	Report writing, poster drafting	3	6	12

Methods and tools. We plan to use Python 3.11, pandas, NumPy, scikit-learn pipelines, XGBoost, LightGBM, CatBoost, and SHAP. Visualization relies on seaborn and matplotlib; documentation lives in Markdown exported to PDF. Kaggle notebooks serve as a backup execution environment for quick experiments.

Comments. Leonid leads the heavier modeling work (T3-T5), Kirsika focuses on data preparation and visualization (T2-T3), and Jessica owns the poster plus final storytelling (T6) while still helping with evaluation. Each column sums to 30 hours so the workload is balanced, and everyone contributes to each milestone with rotating reviewers before submissions and poster deadlines.